A Continuous Model of Computation

Physicists should consider an alternative to the Turing-machine model of computation.

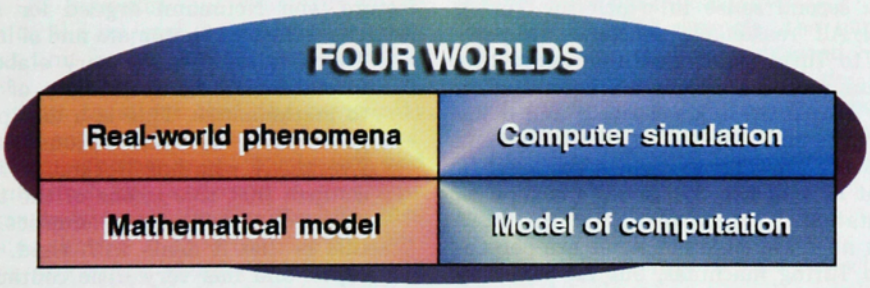
Joseph F. Traub

A central dogma of computer science is that the Turing-machine model is *the* appropriate abstraction of a digital computer. Physicists who've thought about the matter also seem to favor the Turing-machine model. For example, Roger Penrose devoted some 60 pages of a book[1] to a description of this abstract model of computation and its implications. I argue here that physicists should consider the real-number model of computation as more appropriate and useful for scientific computation.

First, I introduce the four "worlds" that play a role here. Above the horizontal line in the diagram on this page are two *real* worlds: the world of physical phenomena and the computer world in which simulations are performed. Below them are represented two *formal* worlds: a mathematical model of some physical phenomenon and a model of computation that is an abstraction of a physical computer. We get to choose both the mathematical model and the model of computation. What type of models should we choose?

The physicist often chooses a continuous mathematical model for the phenomenon under consideration. Continuous models range from the dynamical systems of classical physics to the operator equations and path integrals of quantum mechanics. These models are based on the real numbers (as distinguished from the subset of rational numbers). The real numbers are, of course, an abstraction. It takes an infinite number of bits to represent a single real number. (A rational number, by contrast, requires only a finite number of bits.) But infinitely many bits are not available in the universe. One uses the continuous domain of the real numbers because it is a powerful and useful construct. Let us accept that continuous models are widely used in mathematical physics, and that they will continue to occupy that role for the foreseeable future. But the computer is a finite-state machine. What should we do when the continuous mathematical model meets the finite-state machine?

In the next section I compare and contrast two models of computation: the Turing-machine and real-number models. In the interest of full disclosure, I must tell you that I've always used the real-number model in my work as a computer scientist. But I do my best here to present balanced arguments. Then I show how the real-number model is used in the study of the computational complexity of continuous mathematical models. (Computational complexity is a measure of the minimal computational resources required to solve a mathematically posed problem.) This is the province of a branch of complexity theory called information-based complexity, and what follows is intended to demonstrate the power of this theory.



Two models of computation

Although many readers are familiar with the Turing-machine model I start by describing it briefly. Then, after describing the real-number model, I will discuss the pros and cons of these two models.

Alan Turing, one of the intellectual giants of the twentieth century, defined his machine model to establish the unsolvability of David Hilbert's *Entscheidungsproblem*,[2] the problem of finding an algorithm for deciding (entscheiden, in German) whether any given mathematical proposition is true or false. The Turing machine is a *gedanken*gadget employing a tape of unbounded length, divided into sequential squares, each of which is either blank or contains a single mark. For any particular input, the resulting calculation and output are finite; that is to say, the tape is blank beyond a certain point. The machine's reading head reads one square at a time and, after making or erasing a mark, moves one square to the left or right. The machine has a finite number of internal states. Given its initial state and the input sequence on the tape, the machine changes its state and the head prints a symbol and moves one square. Finally, the machine decides when to halt.

We turn now to a very brief description of the real-number model; a precise formulation may be found in the literature.[3,4] The crux of this model is that one can store and perform arithmetic operations and comparisons on real numbers exactly and at unit cost. (For the moment, I defer discussion of "information operations.")

The real-number model has a long history. Alexander

Joseph Traub *is the Edwin Howard Armstrong Professor of Computer Science at Columbia University in New York City. His homepage is www.cs.columbia.edu/~traub.*

Ostrowski used it in his work on the computational complexity of polynomial evaluation in 1954. In the 1960s, I used the real-number model for research on optimal iteration theory and Shmuel Winograd and Völker Strassen employed it in their work on algebraic complexity. Henryk Woźniakowski and I used the real-number model in a 1980 monograph on information-based complexity. The 1989 formalization of the real-number model for continuous combinatorial complexity by Lenore Blum, Michael Shub, and Steven Smale initiated a surge of research on computation over the real numbers.

Both models are abstractions of real digital computers. Of the Turing-machine model, Penrose wrote, "It is the unlimited nature of the input, calculation space, and output which tells us that we are considering only a mathematical idealization."[1] Which abstraction to use depends on how useful that abstraction is for a given purpose. What are the pros and cons of these two models of computation?

Turing-machine model—pros and cons

In favor of the Turing-machine model, one can say that it's desirable to use a finite-state abstraction for a finite-state machine. Moreover, the Turing machine's simplicity and economy of description are attractive. Furthermore, it is universal in two senses: First is the Church–Turing thesis, which states that what a Turing machine can compute may be considered a universal definition of computability. (Computability on a Turing machine is equivalent to computability in the lambda calculus, a logical system formulated by Alonzo Church in 1936.) Although one cannot prove the Church–Turing thesis, it appeals to our intuitive notion of computability.

"The Turing model is fundamentally inadequate for giving a foundation to the theory of modern scientific computation."

There is also a second sense in which the Turing machine is universal: All "reasonable" machines are polynomially equivalent to Turing machines. Informally, this means that if the minimal time to compute an output on a Turing machine is $T(n)$for an input of size n and if the minimal time to compute an output on any other machine is $S(n)$, then T does not grow faster than a power of S. Therefore, one might as well use the Turing machine as the model of computation. I am, however, not convinced of the assertion that all reasonable machines are polynomially equivalent to Turing machines, but I'll defer my critique for the moment.

What are the disadvantages of the Turing-machine model? I believe it is not natural to use such a discrete model in conjunction with continuous mathematical models. Furthermore, estimated running times on a Turing machine are not predictive of scientific computation on digital computers. One reason for this is that scientific computation is usually done with fixed-precision floating-point arithmetic, so that the cost of arithmetic operations is independent of the size of the operands. Turing-machine operations, by contrast, depend on the sizes of numbers.

Finally, there are interesting computational models that are not polynomially equivalent to the Turing-machine model. Consider the example of a random-access machine in which multiplication is a basic operation and memory access, multiplication, and addition can be performed at unit cost. Such machines go by the ungainly acronym UMRAM. This seems like a reasonable abstraction of a digital computer, in which multiplication and addition on fixed-precision floating point numbers cost about the same. But the UMRAM is *not* polynomially equivalent to a Turing machine! However, a RAM that does not have multiplication as a fundamental operation is polynomially equivalent to a Turing machine.

Real-number model—pros and cons

What are the advantages of the real-number model? Because physicists generally assume a continuum, their mathematical models are often continuous, employing the domain of the real (and complex) numbers. It seems natural, therefore, to use the real numbers in analyzing the numerical solution of continuous models on a digital computer.

If we leave aside the possibility of numerical instability, computational complexity in the real-number model is the same as it is for fixed-precision, floating-point arithmetic. Therefore the real-number model is predictive of running times for scientific computations. Studying computational complexity in the real-number model has led to new, superior methods for doing a variety of scientific calculations.

Another reason for using the real-number model is that it makes available the full power of continuous mathematics. I give an example below, when I discuss a result on non-computable numbers and its possible implications for physical theories. With the real-number model and the techniques of analysis (that is to say, the mathematics of continuous functions), this result is established in about a page. With the Turing-machine model, by contrast, the proof requires a substantial part of a monograph.

The argument for using the power of analysis was already made in 1948 by John von Neumann, one of the leading mathematical physicists of the century and a father of the digital computer. In his Hixon Symposium lecture, von Neumann argued for a "more specifically analytical theory of automata and of information." He said:

> There exists today a very elaborate system of formal logic, and specifically, of logic as applied to mathematics. This is a discipline with many good sides, but also serious weaknesses. . . . Everybody who has worked in formal logic will confirm that this is one of the technically most refractory parts of mathematics. The reason for this is that it deals with rigid, all-or-none concepts, and has very little contact with the continuous concept of the real or of the complex number, that is, with mathematical analysis. Yet analysis is the technically most successful and best-elaborated part of mathematics. . . . The theory of automata, of the digital, all-or-none type as discussed up to now, is certainly a chapter in formal logic.[5]

We may adopt these observations, *mutatis mutandis*, as an argument for the real-number model. Recently, Blum and coauthors[6] have argued for the real-number model, asserting that "the Turing model . . . is fundamentally inadequate for giving . . . a foundation to the theory of modern scientific computation, where most of the algorithms . . . are real-number algorithms."

Against the real-number model, one can point out that digital representations of real numbers do not exist in the real world. Even a single irrational real number is an infinite, nonrepeating decimal that requires infinite resources to represent exactly. We say that the real-number model is not finitistic. But neither is the Turing-machine model, because it utilizes an unbounded tape. It is

therefore potentially infinite. Nevertheless, the Turing-machine model, because it is unbounded but discrete, is less infinite than the real-number model. It would be attractive to have a finite model of computation. There are finite models, such as circuit models and linear bounded automata, but they are only for special-purpose computation.

Information-based complexity

To see the real-number model in action, I indicate below how to formalize computational complexity issues for continuous mathematical problems and then describe a few recent results. To motivate the concepts, I choose the example of d-dimensional integration, because of its importance in fields ranging from physics to finance. I will touch briefly on the case $d = \infty$, that is to say, path integrals.

Information-based complexity assumes that the information is partial, contaminated, and priced.

Suppose we want to compute the integral of a real-valued function f of d variables over the unit cube in d dimensions. Typically, we have to settle for computing a numerical approximation with some error $\varepsilon < 1$. To guarantee an ε-approximation, we need some global information about the integrand. We assume, for example, that this class of integrands has smoothness r. One way of defining such a class is to let F_r be the class of those functions whose derivatives, up through order r, are uniformly bounded.

A real function of a real variable cannot be entered into a digital computer. Therefore, we evaluate the function at a finite number of points, calling that set of values "the information" about f. An algorithm then combines these function values into a number that approximates the integral.

In the worst case, we guarantee an error of at most ε for every f in F_r. The computational complexity is the least cost of computing the integral to within ε for every such f. We charge one unit for every arithmetic operation and comparison, and c units for every function evaluation. Typically, $c \gg 1$. I want to stress that the complexity depends on the problem and on ε, but not on the algorithm. Every possible algorithm, whether or not it is known, and all possible points at which the integrand is evaluated, are permitted to compete when we consider the least cost.

Nikolai Bakhvalov showed in 1959 that the complexity of our integration problem is of order $\varepsilon^{-d/r}$. For $r = 0$, with no continuous derivatives, the complexity is infinite; that is to say, it is impossible to solve the problem to within ε. But even for any positive r, the complexity increases exponentially with d, and we say that the problem is "computationally intractable."

The curse of dimensionality

That kind of intractability is sometimes called the "curse of dimensionality." Very large numbers of dimensions occur in practice. In mathematical finance, d can be the number of months in a 30-year mortgage.

Let us compare our d-dimensional integration problem with the Traveling Salesman Problem, a well-known example of a discrete combinatorial problem. The input is the location of n cities and the desired output is the minimal route that includes them all. The city locations are usually represented by a finite number of bits. Therefore, the input can be exactly entered into a digital computer. The complexity of this combinatorial problem is unknown, but is conjectured to be exponential in the number of cities, rendering the problem computationally intractable. In fact, many other combinatorial problems are conjectured to be intractable. This is a famous unsolved issue in computer science.

Many problems in scientific computation that involve multivariate functions turn out to be intractable in the worst-case setting; their complexity grows exponentially with the number of variables. Among the intractable problems are partial differential and integral equations,[7] nonlinear optimization,[8] nonlinear equations,[9] and function approximation[10].

One can sometimes get around the curse of dimensionality by assuming that the function obeys a more stringent global condition than simply belonging to F_r. If, for example, one assumes that a function and its constraints are convex, then its nonlinear optimization requires only on the order of $\log(1/\varepsilon)$ evaluations of the function.[8]

In general, information-based complexity assumes that the information concerning the mathematical model is partial, contaminated, and priced. In our integration example, the mathematical input is the integrand and the information is a finite set of function values. The information is partial because the integral cannot be recovered from the function values. For a partial differential equation, the mathematical input would be the functions specifying the boundary conditions. Usually, the mathematical input is replaced by a finite number of information operations—for example, functionals on the mathematical input or physical measurements that are fed into a mathematical model. Such information operations,[3,4] in the real-number model, are permitted at cost c.

In addition to being partial, the information is often contaminated,[11] for example, by measurement or rounding errors. If the information is partial or contaminated, one cannot solve the problem exactly. Finally, the information has a price. For example, the information needed for oil-exploration models is obtained by the explosive triggering of shock waves. With the exception of certain finite-dimensional problems, such as finding roots of systems of polynomial equations and doing matrix-algebra calculations, the problems typically encountered in scientific computation have information that is partial, contaminated, and priced.

Information-based complexity theory is developed over abstract spaces such as Banach and Hilbert spaces, and the applications typically involve multivariate functions. We often seek an optimal algorithm—one whose cost is equal or close to the complexity of the problem. Such endeavors have sometimes led to new methods of solution.

The information level

The reason why we can often obtain the complexity and an optimal algorithm for information-based complexity problems is that partial or contaminated information lets one make arguments at the information level. In combinatorial problems, by contrast, this information level does not exist, and we usually have to settle for conjectures and attempts to establish a hierarchy of complexities.

A powerful tool at the information level—one that's not available in discrete models of computation—is the notion of the radius of information, denoted by R. The radius of information measures the intrinsic uncertainty of solving a problem with a given body of information. The smaller this radius, the better the information. An ε-approximation can be computed if, and only if, $R < \varepsilon$.

The radius depends only on the problem being solved and the available information; it is independent of the algorithm. In every information-based complexity setting, one can define an R. (We've already touched on the worst-case setting above, and two additional settings are to come in the next section.) One can use R to define the "value of information," which I believe is preferable, for continuous problems, to Claude Shannon's entropy-based concept of "mutual information."[4]

I present here a small selection of recent advances in the theory of information-based complexity: high-dimensional integration, path integrals, and the unsolvability of ill-posed problems.

Continuous multivariate problems are, in the worst-case setting, typically intractable with regard to dimension. That is to say, their complexity grow exponentially with increasing number of dimensions. One can try in two ways to attempt to break this curse of dimensionality: One can replace the ironclad worst-case ε-guarantee by a weaker stochastic assurance, or one can change the class of mathematical inputs. For high-dimensional integrals, both strategies come into play.

Monte Carlo

Recall that, in the worst-case setting, the complexity of d-dimensional integration is of order $(1/\varepsilon)^{d/r}$. But the expected cost of the Monte Carlo method is of order $(1/\varepsilon)^2$, independent of d. This is equivalent to the common knowledge among physicsists that the error of a Monte Carlo simulation decreases like $n^{-1/2}$. This expression for the cost holds even if $r = 0$. But there is no free lunch. Monte Carlo computation carries only a stochastic assurance of small error.

There is no free lunch. Monte Carlo computation carries only a stochastic assurance of small error.

Another stochastic situation is the average-case setting. Unlike Monte Carlo randomization, this is a deterministic setting with an *a priori* probability measure on the space of mathematical inputs. The guarantee is only that the expected error with respect to this measure is at most ε and that the complexity is the least expected cost.

What is the complexity of integration in the average-case setting and where should the integrand be evaluated? This problem was open for some 20 years until it was solved by Woźniakowski in 1991. Even for the least smooth case $r = 0$, he proved that the average complexity is of order

$$\varepsilon^{-1} (\log \varepsilon^{-1})^{(d-1)/2}$$

and that the integrand should be evaluated at the points of a "low discrepancy sequence." Many such sequences are known in discrepancy theory, a much studied branch of number theory. Roughly speaking, a set of points has low discrepancy if a rather small number of points is distributed as uniformly as possible in the unit cube in d dimensions.

The integral of f is approximated by

$$\frac{1}{n}\sum_{i=1}^{n} f(t_i).$$

In Monte Carlo computations, the points t_i are chosen from a random distribution. If they constitute a low-discrepancy sequence, it is called a quasi–Monte Carlo computation.[12] If we ignore the logarithmic factor in Woźniakowski's theorem, we see that a quasi–Monte Carlo computation costs only $1/\varepsilon$, much less than the classic Monte Carlo method. If d is modest, one can indeed neglect the logarithmic factor. But in financial computations with $d = 360$, it looks ominous.

It was computer experiments by Spassimir Paskov[4] on various financial computations that led to the surprising conclusion that quasi–Monte Carlo always beats ordinary Monte Carlo, often by orders of magnitude. Anargyros Papageorgiou and I have shown that with generalized Faure points, which form a particular kind of low-discrepency sequence, quasi–Monte Carlo computations can achieve accuracies of 1%—which is generally good enough for finance—with only 170 function evaluations.

A recent paper by Woźniakowski and Ian Sloan[13] may explain this surprising power of quasi–Monte Carlo computation. They observe that many problems of mathematical finance are highly non-isotropic; that is to say, some dimensions are much more important than others. This is due, in part, to the discounted value of future money. They prove the existence of a quasi-Monte Carlo method whose worst-case cost is ε^{-p}, independent of d, where p is no bigger than 2.

The quasi–Monte Carlo method also looks promising for problems in areas other than mathematical finance. For example, Papageorgiou and I have reported[14] test results on a model integration problem[15] suggested by integration problems in physics. Once again, quasi–Monte Carlo outperformed traditional Monte Carlo by a large factor, for d as large as 100.

Let me end this discussion with a caveat. There are many problems for which it has been established that randomization does not break intractability. One such instance, as shown by Greg Wasilkowski, is the problem of function approximation.[3] How to characterize those problems for which randomization does breaks intractability remains an open question.

Path integrals

A new area of research of particular interest for physics is the complexity of path integrals.[16] One can define a path integral

$$S(f) = \int_X f(x)\, \mu(\mathrm{d}x),$$

where μ is a probability measure on an infinite-dimensional space X. Path integrals find numerous applications in quantum field theory, continuous financial mathematics, and the solution of partial differential equations. To compute an ε-approximation to a path integral, the usual method is to approximate the infinite-dimensional integral by one of finite dimension, calculated by Monte Carlo.

Here I briefly describe some very recent work on the complexity of path integral computation by Wasilkowski, Woźniakowski, and Leszek Plaskota. They considered a new algorithm for certain Feynman–Kac path integrals that concern the solution of the heat equation

$$\frac{\partial z(u,t)}{\partial t} = \frac{1}{2}\frac{\partial^2 z(u,t)}{\partial u^2} + V(u)\, z(u,t), \quad \text{where } z(u,0) \equiv y(u),$$

with the Feynman–Kac integral

$$z(u, t) = \int_C y(x(t) + u) \exp\left(\int_0^t V(x(s) + u)\, ds\right) w(dx),$$

where V is a potential function, y is an initial condition function, C is the space of continuous functions, and w is its Wiener measure. If V is sufficently smooth and y is constant, Alexandre Chorin's 1973 Monte Carlo algorithm yields a cost of $\varepsilon^{-2.5}$ for stochastic assurance that the error is no more than ε.

By contrast, the new algorithm is deterministic. Its cost is only $\varepsilon^{-0.25}$. And, for that bargain-basement price, one even gets a worst-case guarantee rather than just a stochastic assurance. But the new algorithm does have a drawback. It requires the calculation of certain coefficients given by weighted integrals. So one might say there's a precomputation cost that is not included in the advertised price of the algorithm.

The best is yet to come.

Leaving that aside, $\varepsilon^{-0.25}$ is an upper bound on the complexity of the problem. A lower bound has also been established and, for certain classes, the two bounds are essentially the same. So the complexity of the Feynman–Kac path integral problem is known and the new algorithm is essentially optimal. The results are, in fact, more general than I've indicated here. I just want to give the reader a taste of this work.

Ill-posed problems

As the final example, we look at ill-posed problems from the viewpoint of computability theory and information-based complexity, to show the power of analysis at work. In 1988, Marian Pour-El and Ian Richards[17] established what Penrose has called "a rather startling result." Consider a wave equation for which the data provided at an initial time determines the solution at all later times. Then there exist computable initial conditions with the property that at a later time the field is not computable. A similar result holds for the backward heat equation. Both of these cases are examples of unbounded linear transformations, and they illustrate a theorem that tells us that an unbounded linear transformation can take computable inputs into non-computable outputs.

Pour-El and Richards devoted a large part of their monograph to proving this theorem by means of computability theory. An analogous result was established by Arthur Werschulz,[18] by information-based complexity theory over the real numbers. Werschulz's proof, which is only about a page long, illustrates the power of analysis. As we shall see, his approach has several other advantages.

The classical theory of ill-posed problems was developed by Jacques Hadamard a century ago. Roughly speaking, a problem is ill-posed if small changes in the problem's specification can cause large changes in the solution. Hadamard felt there was something inherently wrong with trying to solve such problems. Given an ill-posed problem, one should attempt to reformulate it. There are, however, many important ill-posed problems that must be confronted in practice—for example, in remote sensing and computational vision.

Suppose we want to compute Lf, where L is a linear operator on the function f. The operator is said to be unbounded if, roughly speaking, the size of Lf is arbitrarily bigger than that of f, for some f. It is well known that Lf is ill-posed if, and only if, L is unbounded. The backward heat equation and the wave equation with an initial condition that is not twice differentiable are examples of problems whose solution is given by an unbounded linear operator. Werschulz assumed that the function f (which, for a differential equation, might be the initial condition) cannot be entered into a digital computer. So he discretized f by evaluating it at a finite number of points. Werschulz proved that, if the problem is ill-posed, it is impossible to compute an ε-approximation to the solution at finite cost for any ε, no matter how large. Thus the problem is *unsolvable*, which is a much stronger result than mere noncomputability.

But the best is yet to come. The notion of an unbounded linear operator is a worst-case concept. In information-based complexity, however, it is natural to consider average behavior. Werschulz places a measure on the inputs—for example, the initial conditions. He says a problem is well-posed, *on average*, if the expectation of Lf with respect to this measure is finite. Then it can be shown that every ill-posed problem is well-posed, on average, for every Gaussian measure and therefore solvable, on average.[4] So we see that the unsolvability of ill-posed problems is a worst-case phenomenon. It melts away, on average, for reasonable measures.

I've tried here to provide a taste of some of the achievements of information-based complexity theory with the real-number model of computation. I invite the reader to look at some of the monographs cited in the references for other results and numerous open problems.

My research reported here was supported in part by the National Science Foundation and the Alfred P. Sloan Foundation. I appreciated the comments of Jerry Altzman, Anargyros Papageorgiou, Lee Segel, Arthur Werschulz, and Henryk Woźniakowski on the manuscript.

References

1. R. Penrose, The Emperor's New Mind, Oxford U. P., Oxford (1989).
2. A.M. Turing, Proc. London Math. Soc. **42**, 230 (1937).
3. J. F. Traub, G. W. Wasilkowski, H. Woźniakowski, *Information-Based Complexity*, Academic Press, New York (1988).
4. J. F. Traub, A. G. Werschulz,, *Complexity and Information*, Cambridge U. P., Cambridge, England (1998).
5. J. von Neumann, *Collected Works*, V. A. Taub, ed., Macmillan, New York (1963).
6. L. Blum, F. Cucker, M. Shub, S. Smale, *Complexity and Real Computation*, Springer-Verlag, New York (1998).
7. A. G. Werschulz, *The Computational Complexity of Differential and Integral Equations: An Information-Based Approach*, Oxford U. P., Oxford (1991).
8. A. S. Nemirovski, D. B. Yudin, *Problem Complexity and Method Efficiency in Optimization*, Wiley-Interscience, New York (1983).
9. K. Sikorski, *Optimal Solution of Nonlinear Equations*, Oxford U. P., Oxford (1999).
10. E. Novak, *Deterministic and Stochastic Error Bound in Numerical Analysis*, Lecture Notes in Mathematics, No. 1349, Springer-Verlag, New York (1988).
11. L. Plaskota, *Noisy Information and Computational Complexity*, Cambridge U. P., Cambridge, England (1996).
12. S. Paskov, J. F. Traub, J. Portfolio Management, **22**, 113 (1995).
13. I. Sloan, H. Woźniakowski, J. Complexity **14**, 1 (1998).
14. A. Papageorgiou, J. F. Traub, Computers in Physics **11**, 574 (1997).
15. B. Keister, Computers in Physics **10**, 119 (1996).
16. G. W. Wasilkowski, H. Woźniakowski, J. Math. Phys. **37**, 2071 (1996).
17. M. B. Pour-El, J. I. Richards, *Computability in Analysis and Physics*, Springer-Verlag, New York (1988).
18. A. G. Werschulz, Numer. Func. Anal. **9**, 945 (1987). ■